Selective Rendering: Computing only what you see

Alan Chalmers* University of Bristol, UK
Kurt Debattista† University of Bristol, UK
Luis Paulo dos Santos‡ University of Minho, Portugal

Abstract

The computational requirements of a full physically-based global illumination solution are significant, currently precluding its solution on even a powerful modern PC in reasonable let alone real time. A key factor to consider if we are ever to achieve so-called "Realism in Real-Time", is that we are computing images for humans to look at. Although the human visual system is very good, it is by no means perfect. By understanding what the human does, or perhaps more importantly, does not see, enables us to save significant computation effort without any loss of perceptual quality of the resultant image. This paper describes the novel techniques of *selective rendering* which allow us to direct computational resources to those areas of high perceptual importance while avoiding computing any detail which will not be perceived by the viewer. Such selective rendering methods offer us the real possibility of achieving high fidelity graphics of complex scenes at interactive rates.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.3.2 [Computer Graphics]: Graphics Systems

Keywords: Global illumination, high-fidelity graphics, realism in real-time, selective rendering, there-reality

1 Introduction

Realism in Real-Time, computing physically-based global illumination graphics at interactive rates, has been a much sought after goal in computer graphics research for many years. The complexity of modelling a real scene are such that, even with the advent of modern, powerful GPUs, it is simply not possible to render high-fidelity images of such scenes in reasonable, let alone real-time. If, however, we consider that we are rendering images for humans to view, then we can exploit knowledge of the human visual system (HVS), for although the HVS is good, it is not perfect.

When viewing a scene our visual attention is a series of conscious and unconscious processes which enables us to find and focus on relevant parts of the scene quickly and efficiently. Human eyes do not scan a scene in a raster-like fashion, but rather our eyes jump rapidly between features within the scene, so-called saccades. Visual attention comprises two processes, bottom-up and top-down [James 1890].

The bottom-up process is purely stimulus driven. This is evolutionary, attracting our eyes without volitional control to salient features in a scene, for example movement, which may be a predator lurking in a bush, or the red apple in a green tree, indicating that the fruit is ripe. A *saliency map* may calculated for any image, highlighting the perceptually important features [Itti et al. 1998]. The top-down process, on the other hand, is under volitionary control, with the HVS over-riding the bottom-up process and focussing attention on one more objects that are relevant to accomplish a specific visual task, for example, looking for a lost child, or counting the number of occurrences of a certain object. When conducting a task within a scene, even conspicuous objects in a scene that would normally attract the viewers attention are ignored if they are not relevant to the task at hand. This is known as Inattentional Blindness [Mack and Rock 1998].

There-Reality environments have been defined as virtual environments in which the perceptual response of viewers is the *same* as the response they would have if they were present, that is *there*, in the equivalent real environment [Chalmers et al. 2006b]. In this paper, we will show how *selective rendering* techniques, which allow us to direct computational resources to those areas of high perceptual importance while avoiding computing any detail which will not be perceived by the viewer, offer the real possibility of achieving such there-reality environments at interactive rates.

The rest of the paper is organised as follows. Section 2 gives a brief overview of the previous work in selective rendering. Section 3 presents three different selective renderers and shows and their potential for significantly reducing overall computation time. Finally, Section 4 presents some conclusions and discusses how selective rendering has the opportunity, perhaps coupled with parallel rendering, to help us reach the goal of "Realism in Real-Time". Further details of these, and other similar, techniques can be found in [Debattista 2006]

2 Background

The principles of selective rendering have previously been applied in computer graphics in three general ways. Adaptive techniques concentrate computational resources on those areas of an image which are deemed currently most important, according to some criteria. Incremental techniques, on the other hand, progressively refine an image until some stopping condition is met. Finally, component based approaches allow the system to control rendering at the component level, for example, the specular or transparent components. These components will be computed only if, for example in a time-constrained system, sufficient time remains. Selective rendering algorithms can thus be defined as those techniques which require a number of rendering quality decisions to be taken and acted upon prior to, or even dynamically during the actual computation of any image or frame of an animation [Debattista 2006].

The work by Clark in the 1970s was amongst the first to consider some form of selective rendering related to level of detail [Clark 1976]. Since then selective rendering for rasterisation approaches have lowered computational costs by reducing the number of geometrical objects which are sent to the rendering pipeline. Much of this recent work is summarised in [Luebke et al. 2002]. Bergman et al. took a different approach. Here the complexity of the shading is selectively refined using a set of heuristics [Bergman et al. 1986].

*e-mail: alan.chalmers@bris.ac.uk
†e-mail:debattista@cs.bris.ac.uk
‡e-mail:psantos@di.uminho.pt

GRAPHITE 2006, Kuala Lumpur, Malaysia, November 29 – December 02, 2006.
© 2006 ACM 1-59593-564-9/06/0011 $5.00

Within the field of global illumination, progressive refinement radiosity [Cohen et al. 1988] provided the viewer with an approximate image which was refined over time. An overview of other perceptually driven radiosity algorithms is given in [Prikryl and Purgathofer 1999]. Whitted's original ray tracer included an selective antialising technique where further rays were only shot within a pixel if the intensity at pixel corners was below a predefined threshold [Whitted 1980]. Since then Mitchell developed adaptive antialiasing for ray tracing [Mitchell 1987], Painter and Sloan presented an adaptive progressive refinement ray tracer [Painter and Sloan 1989], and in 1991 Chen et al. presented a progressive refinement algorithm for a multipass rendering algorithm combining progressive radiosity, ray tracing and backwards ray tracing, where the individual passes were interruptible [Chen et al. 1991].

In 1998 Myszkowski introduced the visual difference predictor to global illumination [Myszkowski 1998]. Based on Dalys VDP [Daly 1993], Myszkowski's VDP was used to selectively stop rendering in a Monte Carlo path tracer based on the perceptual difference between images at regular intervals. Since this pioneering research, a growing body of work has appeared on using visual perception in computer graphics. Bolin and Meyer used a visual difference predictor both to direct the next set of samples within a stochastic ray tracing framework and as a stopping condition [Bolin and Meyer 1998]. Ramasubramanian et al. [Ramasubramanian et al. 1999], Haber et al. [Haber et al. 2001] and Yee et al. [Yee et al. 2001] all exploited knowledge of the salient parts of a scene to render these at high quality, while the remainder could be computed at a lower quality without the viewer being aware of this quality difference. Cater et al. [Cater et al. 2002; Cater et al. 2003] and subsequently Sundstedt et al. [Sundstedt et al. 2004] extended this to also consider the visual task a user may be performing in a scene. By only computing at high quality those parts of the scene that the viewer is actually attending to while performing the task, significant computational time was saved with out any loss in perceptual quality being experienced by the user. A good overview of perceptually adaptive graphics can be found in [O'Sullivan et al. 2004].

In the case of component based rendering, Wallace et al. presented a multipass algorithm that computed the diffuse component with a rendering pass and used a z-buffer algorithm for view dependent planar reflections [Wallace et al. 1987]. The view independent nature of the indirect diffuse component of global illumination was exploited by Ward et al. to sufficiently populate the innovative irradiance cache, from which other samples could interpolate with significant computational cost savings [Ward et al. 1988]. Other component based rendering have included [Sillion and Puech 1989; Shirley 1990; Heckbert 1990; Chen et al. 1991; Stokes et al. 2004]. In 1998, Slusallek et al. introduced the concept of lighting networks to render scenes based on the users combining the implementations of different rendering algorithms, typically radiosity and ray tracing techniques, into a network [Slusallek et al. 1998].

In 2005, Debattista et al. introduced a component-based selective rendering system in which the quality of every image, and indeed every pixel, was controlled by means of a component regular expression (crex). This crex, inspired by Heckbert's light transport notation [Heckbert 1990], provided a flexible mechanism for controlling which components are rendered and in which order. The crex can thus can be used as a strategy for directing the light transport within a scene and also in a progressive rendering framework.

3 Selective Renderers

Selective rendering is thus the technique of directing computational efforts to those parts of the rendering process which are deemed currently most important according to some predefined, or dynamically acquired, information. In this section we present three selective renderers which exploit different approaches to determine where best to direct the computation.

3.1 Detecting Key Objects in a Frame

This dynamic approach enables the selective renderer to detect the presence of predefined key objects, for example an on-screen distractor (OSD), such as a sound-emitting object, in any frame of animation as an integral part of the rendering process. When one or more OSDs are present, these are rendered in high quality, while the remainder of the frame can be rendered at a much lower quality without the viewer being aware of this quality difference [Cater et al. 2003; Mastoropoulou et al. 2005]. This selective renderer can be used for both top-down and bottom-up visual attention processes, that is, whether an object on the screen draws the viewer's attention as part of a task or involuntarily.

The first phase of this selective renderer involves rendering to a base quality level, typically 1 ray per pixel, while locating the OSDs. The second phase renders the image selectively applying a foveal angle gradient decrease in quality around the projected image of the detected OSDs and maintaining quality consistency using a quality buffer, *q-buffer*. Rays per pixel are used to modulate the rendering quality. Since this selective renderer is primarily created for animations, an animation file maintains a list of all frames with the default rendering quality for each frame and whether any OSD should be considered more salient. Frames that contain distractors are tagged as selective. We also need the option of not having objects as always being distractors, but perhaps being distractors only for certain frames. This is particularly useful for sound emitting objects which may only be OSDs when they are actually emitting sounds [Mastoropoulou et al. 2005]. Those frames which do not contain any OSDs are rendered at the default high quality.

When rendering the frames tagged as selective in an animation, the selective rendering process can be viewed as a two pass process for each frame. In the first pass, the image is rendered using the traditional rendering method up to the base quality. In this pass the distractor objects are detected through object identification of the primary ray or certain class of secondary rays, as they intersect the distractors' geometry. We term the rays that are allowed to detect the distractors *detectable rays*. Only certain types of secondary rays are detectable rays, such as pure specular which represent pure reflections of the object. Other secondary rays, for example, indirect diffuse rays, would reflect very little of the true nature of the distractor. A data structure termed an on-screen distractor list (OSDL) is a queue data structure responsible for storing OSD object data and pixel locations. When the intersection routine for the detectable rays returns the object hit, a distractor object list is consulted and if the object appears on the list, the OSDL adds the detectable object to the list. The first phase ends when the image is rendered entirely to base quality, at which point the computation of the OSDL is completed for the first phase. The OSDL can be visualised as an on-screen distractor map. OSD maps for a number of scenes are shown in Figure 1.

The second phase introduces another data structure, the quality buffer (q-buffer) which ensures that the correct number of rays are shot for each pixel. The q-buffer is inspired by the z-buffer used for

solving depth issues in hidden surface removal of rasterised rendering [Catmull 1978] and is a buffer equal in size to the resolution of the image to be rendered. It stores the quality in rays per pixel of the number of rays rendered up to that point for that given pixel. At the beginning of the second phase all entries in the q-buffer are initialised to the value of the base quality. In the second phase, the OSDL is parsed and for each entry a number of rays intended to be shot is calculated. The ray's hit point is also tested to discover whether or not the pixel is a boundary pixel of the projected image of the distractor object. If there are no distractor objects at this point, no further action is taken. If the pixel corresponds to an internal point on the distractor's projected image, the difference between the distractor's value and the q-buffer at that pixel is calculated. If this difference is positive a number of primary rays equal to the difference is shot. The q-buffer at that pixel is then set equal to the distractor's quality value. If the difference is negative or zero, no action is taken.

For border pixels, the renderer degrades the quality around the border of the distractor objects within a user-defined radius, usually the foveal angle. This option provides a method of rendering around the foveal angle that is not limited to the size of the object. Each pixel within this radius is cycled through and, for each pixel, the desired quality in rays per pixel is calculated. The q-buffer is then consulted and if the new quality is greater than the corresponding entry in the q-buffer the difference in rays is shot and the q-buffer entry updated in the same method as described above. The q-buffer is necessary to protect the quality of pixels lying within the degradation radius of more than one different objects which might result in more rays than necessary contributing to a given pixel. As the rendering progresses, the image is refined so the projected distractor objects on the image plane's boundaries might change subtly. This algorithm ensures that the new boundaries are updated accordingly by storing the new information onto the OSDL. The q-buffers for various OSD maps and scenes can be seen in figure 1 (right).

3.1.1 Results

This selective renderer was implemented as an extension to the light simulation software Radiance [Ward 1994]. A number of user adjustable parameters were added. Firstly, a base quality and a high quality is set for each frame. Secondly, for animations, a frame needs to be tagged as selective or high quality, and if selective, a list of distractor objects for that frame is identified. If a frame is tagged as high quality the entire image is rendered in high quality. This option is used when no distractors are present. For frames that are tagged as selective, the base quality is rendered in the first phase and the selective quality in the second phase, following the algorithm described above. Note that some frames might not have distractor objects and still be marked as selective; useful for when there is a distractor outside the view images, such as an abrupt sound. The results show, for a number of scenes, how the presence of OSDs may be exploited to significantly reduce overall computation time.

In Figure 1, the distractors are: the exit signs for the Corridor scene and the glossy sphere for the Cornell Box. For the corridor scene the rendering parameters do not detect the distractor in the reflections. For the Cornell Box scene the rendering parameters were set to modulate the importance depending on the material of the object in the reflections. The differences can be clearly seen in the q-buffered OSD maps of the Cornell box scene. Rendering parameters were set to default settings with a maximum of 16 rays per pixel, and a base quality of 1 ray per pixel. The irradiance cache was pre-computed.

Results, in seconds, to render the images are shown in Table 1. The results demonstrate a significant speedup for some of the renderings and lesser speedup for others. This was expected since the object coverage is both object and view dependent.

	Corridor	Cornell Box
gold	2,454	179
selective	335	50
speedup	7.32	3.58

Table 1: Results for scenes used with the on-screen distractors selective renderers. Timing in seconds.

3.2 Selective Component-Based Rendering

Previous work on selective rendering has predominantly determined quality as a function of number of rays traced, since rays per pixel was the only selective variable. The more salient a pixel, the more rays were traced for that pixel as in [Cater et al. 2003; Sundstedt et al. 2004] and most of the selective renderers discussed in the previous sections. In Yee et al.'s work the saliency effected only the indirect diffuse computation, in particular the accuracy of the search radius used in examining previously cached samples inside the irradiance cache [Yee et al. 2001]. Our approach extends this notion by empowering the renderer with the ability to terminate the path of a ray at any point of its execution as dictated by a component regular expression termed the *crex* [Debattista et al. 2005].

Inspired by Heckbert's light transport notation [Heckbert 1990], the component regular expression, or *crex* , informs the renderer on the order in which the components are to be rendered. The *crex* takes the form of a string of characters with each character representing either a component or a special character used for recurrence or grouping, as shown in Table 2. The BNF of the syntax is presented in Table 3. The alphabetic characters each represent an individual component. The order of the components in the *crex* dictates the order in which the components are rendered. Components spawn rays to be executed by subsequent components or groups. The integer multiplier is used to execute a component or group k times. The $*$ operator is used to execute the previous component or group until no more rays belonging to the recursed components are spawned. The groups ( ) and < > are used to group components together. When using ( ) the components or groups of components within a group inherit the spawned rays from the previous components within the group. On the other hand when using < > all of the rays spawned within the group will be executed when the < > block terminates. The components within < > can be executed in parallel.

In this form of selective rendering, the quality is controlled by the components rather than the more traditional rays per pixel. The initial rendering stage renders a few of the components and then generates an importance map [Sundstedt et al. 2005]. The part of the *crex* which is grouped in { } is modulated by the value in the importance map for a given pixel. No recursion ($*$) is allowed in { }, but different { } can be separated. The components in { } are ordered by importance such that the first components require a lesser value in the importance map to be rendered. Effectively quality of this selective renderer is dictated by the number of component rays shot. The importance map for the rendered image seen in Figure 2 (right) can be seen in Figure 3 (middle). The importance map in this case is based on the task only. Figure 3 (right) shows a colour-coded visualisation of how the *crex* effects the individual pixels for the rendered image. The part of the *crex* prior to the first { is used as pre-selective guidance. Further examples of the entire selective rendering pipeline can be seen in Figure 4. For example, the Desk

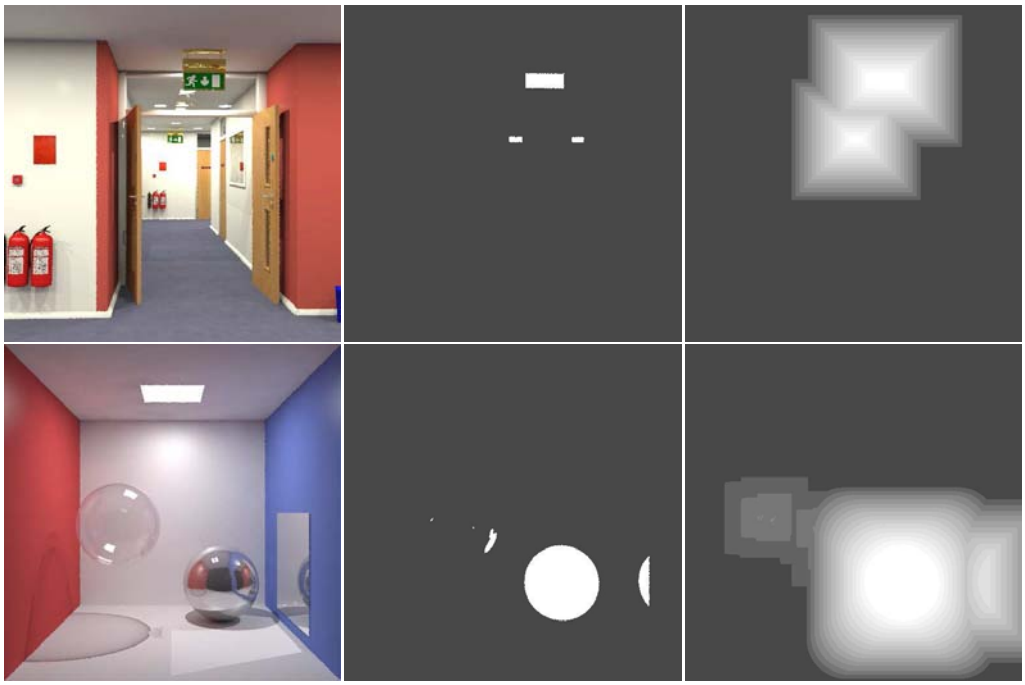

Figure 1: Selective rendering of key objects. From top to bottom: the Corridor scene and the Cornell Box scene detecting OSDs with modulation for the reflected on-screen distractors. From left to right: the rendered scenes, the OSD maps and the q-buffers.

Character	Description
( )	Group one or more component. The latter components in the group execute rays spawned by the former in the group.
< >	Group one or more component. Any spawned ray is never launched within the group but is executed after the group terminates.
{ }	Group one or more component. Group in { } is modulated by an importance map.
k positive integer	Execute last component or group *k* times.
*	Execute until no more rays spawned.
D	Indirect diffuse.
S	Indirect specular.
G	Indirect glossy.
T	Transmitted glass/dielectric†.
R	Reflected glass/dielectric†.
M	Mirror†.

Table 2: The component regular expression description (*crex*). † Shader specific component.

scene is initially rendered with a *crex* of TTM from which the importance map is generated and the modulated using the components in the { } part of the full *crex* of TTM{RSGDTTRSG}.

3.2.1 Results

In order to demonstrate the effectiveness of component-based rendering while exploiting visual attention we ran a task-based psychophysical experiment similar to [Cater et al. 2003] except that in our case the quality is determined by the *crex* rather than the resolution. The full high quality rendering shown in Figure 2 took one hour while the selectively rendered image only required half an hour using a *crex* of T{RSGM}.

The following results only use saliency maps. Also, since we are limiting ourselves only to component-based rendering, all rendering is performed with one ray per pixel at a resolution of 512×512. We use scenes shown in Figure 4 . For rendering the Desk scene the *crex* used was TTM{RSGDTTRSG}, while for the Corridor scene the *crex* was TT{TTMRSGD}2. Results for the component based rendering (cbr), shown in Table 4, demonstrate reasonable speedup compared to the full (gold) rendering times.

	Desk	Corridor
gold	326	712
cbr	86	163
speedup	3.79	4.37

Table 4: Speedup for the selective component-based renderer. Timing in seconds.

We may now compare selective rendering between a traditional rays-per-pixel selective renderer and the selective component-based render using time-constrained rendering systems [Debattista 2006] to fix the time allowed for each renderer to compute. For the traditional selective renderer a maximum of 16 rays per pixel is used, while a fixed 4 fixed rays per pixel is set for the selective component-based renderer. A time constraint of 690 seconds, equivalent to half the time it took to render the reference image, was used. The component-based renderer uses a *crex* of MT{TRSGD}3.

Resulting images and the reference image are shown in Figure 5. The results demonstrate that the component-based time-constrained

< *crex* >	::=	**(** < *crex* > **)** | **<** < *crex* > **>** | **{** < *crex* > **}** | **[** < *crex* > **]** | < *component* > | < *crex* >< *component* > | < *crex* >< *mult* >
< *component* >	::=	**D** | **G** | **S** | **T**‡ | **R**‡ | **M**‡
< *digit* >	::=	**0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**
< *integer* >	::=	< *digit* > | < *digit* >< *integer* >
< *mult* >	::=	∗ | < *integer* >

Table 3: *crex* BNF. ‡ Implementation specific for our RADIANCE-based version.

Figure 2: One set of images from the Corridor scene used for the visual attention experiment: (left) high quality image (HQ) and (right) component-based quality (CBQ).

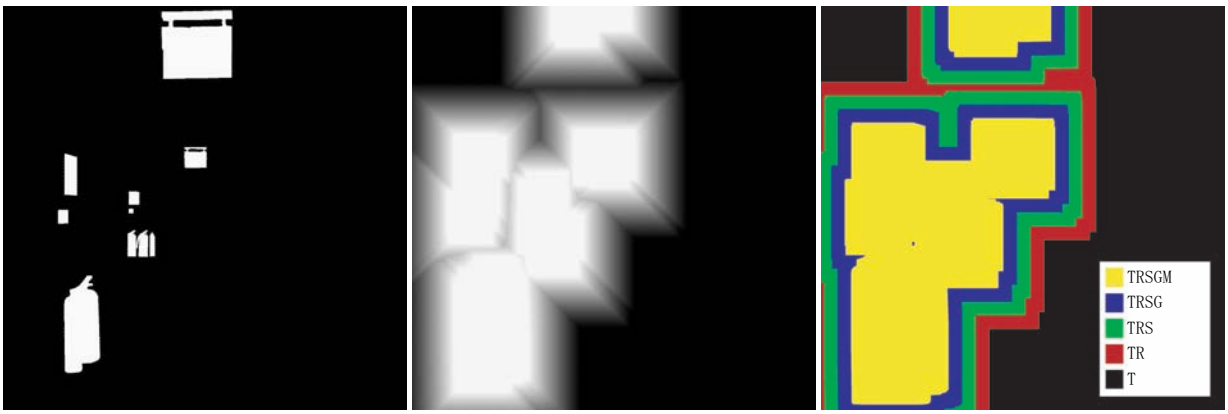

Figure 3: A visualisation of the importance map: (left) the task map (HQ), (middle) the task map with foveal angle gradient and (right) a colour-coded visualisation of which components of the *crex* are rendered for each pixel for a *crex* of `T{RSGM}`.

renderers scale better to lower time constraints since the scheduling and profiling is at a finer grain than that of the traditional time-constrained renderer. Further experiments may be needed for conclusive evidence.

3.3 Parallel Selective Rendering

Finally we present a selective parallel rendering framework and demonstrate how it is possible to significantly reduce rendering times by exploiting these approaches to near real-time high-fidelity rendering for complex scenes [Chalmers et al. 2006a]. We demonstrate how selective rendering can make use of various hardware in particular, distributed systems and graphics hardware to achieve significant performance improvements.

The selective rendering framework is shown in Figure 6. The first stage involves generating an image preview using rapid rasterisation rendering. The guidance for the selective rendering is based on an importance map, which can be composed of many different maps. Many of these can be generated using graphics hardware. The final selective rendering stage utilises parallel computing to improve rendering times.

Figure 4: The Desk and Corridor scenes. From left to right and top to bottom: pre-selective rendering images, the selective guidance, selective component-based images and the traditionally rendered image.

Figure 5: Comparisons between the time-constrained renderers for the Desk scene. Selective component-based time-constrained rendering (top left) and selective time-constrained rendering (top right). The reference image (below).

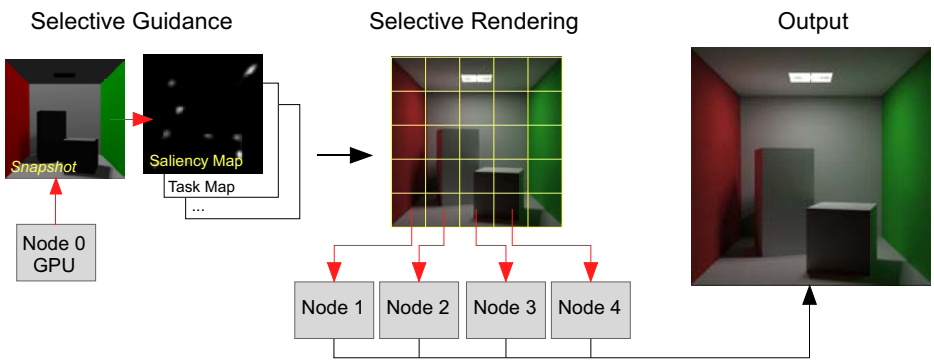

Figure 6: Rendering on demand framework.

Figure 7: The Corridor scene (top) and Tables scene (bottom). Left to right: the rapid image estimate, the saliency map and the selectively rendered final image.

We use the *Snapshot* technique to produce, using graphics hardware, a rapid image estimate of the scene to be rendered [Longhurst et al. 2005a]. A saliency map may be generated from Snapshot using object space knowledge to identify conspicuity for a number of features: motion similar to the work of [Yee et al. 2001], depth and habituation, a novel feature which accounts for the amount of time that an object has been on screen in an animation. Image space saliency, calculated in a similar fashion to the Itti at al. saliency map [Itti et al. 1998], is composed of three channels: colour, intensity and orientation. All of these features are calculated on the GPU for maximum performance, in the order of tens of milliseconds. Finally the Snapshot is used to locate task related object, tagged at a modelling stage, and thus rapidly allows the importance map to be generated[Longhurst et al. 2005b; Longhurst et al. 2006]. Figure 7 shows some scenes rendered using the Snapshot, the generated importance map, which in these cases is only a saliency map, and the final rendering.

In this parallel selective renderer, the selective rendering process uses parallel processing to speed up the rendering even further. Also the quality is not only modulated by rays per pixel but also by a separate component, the irradiance cache search radius [Yee et al. 2001], which improves rendering times substantially for rendering scenes without a pre-computed irradiance cache. The importance map is used by the master process to subdivide the workload and by the slave processes to decide how to modulate the rendering parameters. The master is responsible for subdividing the image plane into a number of tiles of a given granularity. Each image tile represents a job for a slave to compute. The importance map is used as a simple cost prediction map. Since, at the slave, the importance map dictates the number of rays shot per pixel, the master uses it to improve subdivision by ensuring that each tile contains an equal number of primary rays to be shot. This improves load balancing by ensuring a more even distribution of the workload. The improvement is of 2% to 4% in terms of computation time when compared to a fixed tile demand driven approach. Although the computational requirements of each individual ray may differ, the demand driven approach together with the subdivision map alleviates the problem significantly.

The master farms out the work to all the slaves in the form of the coordinates of the tile to be rendered. The slaves then render the image tile assigned to them using the importance map to direct the rendering of each pixel. The sole selective variable in this case is rays per pixel. When the slave finishes executing the job, it asks for more work from the master until the entire process is completed.

3.3.1 Results

Ray tracing is traditionally easily extended into a parallel framework, however our approach follows the Radiance implementation. Although Radiance uses distributed ray tracing to render images, the irradiance cache [Ward et al. 1988] is used to accelerate the calculation of the indirect diffuse component. As the irradiance cache is a shared data structure, it is non-trivial to parallelise.

As with the previous selective renderers, this parallel approach was implemented in Radiance. Distributed computation over a network of workstations was performed using the MPI message passing protocol. In addition, the irradiance cache ws parallelised using the component-based parallel irradiance cache [Debattista et al. 2006], which substantially improves parallel performance by subdividing the indirect diffuse computation from the rest of the components. Results rendered on a cluster of 16 2.4 GHz processors with 3GB RAM are presented in Table 5. As can be seen the speedup is significant overall compared to the traditional uniprocessor computation.

	Cornell Box		Tables		Corridor	
	Time	Tsu	Time	Tsu	Time	Tsu
gold	450	1	4,057	1	4,500	1
selective	130	3.46	1,793	2.26	1,859	2.42
2	62	7.26	832	4.88	882	5.10
4	34	13.24	457	8.88	451	9.98
8	17	26.47	249	16.29	234	19.23
16	11	40.91	149	27.23	141	31.91

Table 5: Timings in seconds for the rendering on demand selective renderer. Tsu for total speedup resulting from both parallelism and selective rendering.

4 Conclusions

A key factor in striving for "Realism in Real-Time" is to realise that we are computing high-fidelity images for humans, and while the human visual system is very good it is not perfect. Exploiting knowledge of the human visual system enables us to selectively render only parts of a scene at the highest quality, and the remainder of the scene at a significantly lower quality, and thus much less computational cost, without the viewer being aware of this difference in quality. In this paper we have discussed three different selective rendering approaches. As we can see from the final method, when selective rendering is combined with parallel processing we are able to achieve significant speedups on a modest number of processors, for example a speed up of over 40 on 16 processors for the Cornell box scene. Future work will look at combining the different selective rendering methods into a single selective rendering framework capable of adopting the most appropriate approach depending on the scenes being rendered. In addition, more use will be made of the specialist hardware present at each node of a modern cluster, for example the presence of one or more GPUs, to even further improve the performance, perhaps finally making "Realism in Real-Time" a reality.

5 Acknowledgements

The work presented in this keynote paper is based largely on the research of Debattista's PhD thesis [Debattista 2006]. We would also like to thank Veronica Sundstedt for the use of the corridor model.

References

BERGMAN, L., FUCHS, H., GRANT, E., AND SPACH, S. 1986. Image rendering by adaptive refinement. In *SIGGRAPH '86*, ACM Press, 29–37.

BOLIN, M. R., AND MEYER, G. W. 1998. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98*, ACM Press, 299–309.

CATER, K., CHALMERS, A., AND LEDDA, P. 2002. Selective quality rendering by exploiting human inattentional blindness: looking but not seeing. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM Press, 17–24.

CATER, K., CHALMERS, A., AND WARD, G. 2003. Detail to Attention: Exploiting Visual Tasks for Selective Rendering. In *Proceedings of the Eurographics Symposium on Rendering*, 270–280.

CATMULL, E. 1978. A hidden-surface algorithm with anti-aliasing. In *SIGGRAPH '78*, ACM Press, 6–11.

CHALMERS, A., DEBATTISTA, K., GILLIBRAND, R., LONGHURST, P., AND SUNDSTEDT, V. 2006. Rendering on demand. In *EGPGV2006 - 6th Eurographics Symposium on Parallel Graphics Visualization*, Eurographics, 9–18.

CHALMERS, A., DEBATTISTA, K., GILLIBRAND, R., LONGHURST, P., SUNDSTEDT, V., AND MASTOROPOULOU, G. 2006. There-reality: Interactive high-fidelity virtual environments. *Parallel Computing*. to appear.

CHEN, S. E., RUSHMEIER, H. E., MILLER, G., AND TURNER, D. 1991. A progressive multi-pass method for global illumination. In *SIGGRAPH '91*, ACM Press, 165–174.

CLARK, J. H. 1976. Hierarchical geometric models for visible surface algorithms. *Commun. ACM 19*, 10, 547–554.

COHEN, M. F., CHEN, S. E., WALLACE, J. R., AND GREENBERG, D. P. 1988. A progressive refinement approach to fast radiosity image generation. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, ACM Press, 75–84.

DALY, S. 1993. The Visible Differences Predictor: An Algorithm for the Assessment of Image Fidelity. In *Digital Images and Human Vision*, A.B. Watson, MIT Press, Cambridge, MA, 179–206.

DEBATTISTA, K., SUNDSTEDT, V., SANTOS, L., AND CHALMERS, A. 2005. Selective component-based rendering. In *Proceedings of GRAPHITE 2005, ACM SIGGRAPH*.

DEBATTISTA, K., SANTOS, L. P., AND CHALMERS, A. 2006. Accelerating the irradiance cache through parallel component-based rendering. In *EGPGV2006 - 6th Eurographics Symposium on Parallel Graphics Visualization*, Eurographics, 27–34.

DEBATTISTA, K. 2006. *Selective Rendering for High-Fidelity Graphcs*. PhD Thesis, University of Bristol.

HABER, J., MYSZKOWSKI, K., YAMAUCHI, H., AND SEIDEL, H.-P. 2001. Perceptually guided corrective splatting. In *Computer Graphics Forum*, vol. 20, 142–152.

HECKBERT, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, ACM Press, 145–154.

ITTI, L., KOCH, C., AND NIEBUR, E. 1998. A model of Saliency-Based Visual Attention for Rapid Scene Analysis. In *Pattern Analysis and Machine Intelligence*, vol. 20, 1254–1259.

JAMES, W. 1890. A saliency-based search mechanism for overt and covert shifts of visual attention. In *Principles of Psychology*.

LONGHURST, P., DEBATTISTA, K., AND CHALMERS, A. 2005. Snapshot: A rapid technique for driving a selective global illumination renderer. In *WSCG 2005 SHORT papers proceedings*.

LONGHURST, P., DEBATTISTA, K., GILLIBRAND, R., AND CHALMERS, A. 2005. Analytic antialiasing for selective high fidelity rendering. In *Proceedings of SIBGRAPI, IEE Computer Society Press*, 359–366.

LONGHURST, P., DEBATTISTA, K., AND CHALMERS, A. 2006. A gpu based saliency map for high-fidelity selective rendering. In *Proceedings of AFRIGRAPH 2006, ACM SIGGRAPH*, 21–29.

LUEBKE, D., WATSON, B., COHEN, J. D., REDDY, M., AND VARSHNEY, A. 2002. *Level of Detail for 3D Graphics*. Elsevier Science Inc.

MACK, A., AND ROCK, I. 1998. *Inattentional Blindness*. MIT Press.

MASTOROPOULOU, G., DEBATTISTA, K., CHALMERS, A., AND TROSCIANKO, T. 2005. Auditory bias of visual attention for perceptually-guided selective rendering of animations. In *Proceedings of GRAPHITE 2005, ACM SIGGRAPH*.

MITCHELL, D. P. 1987. Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, 65–72.

MYSZKOWSKI, K. 1998. The Visible Differences Predictor: Applications to global illumination problems. In *Eurographics Workshop on Rendering*, 223–236.

O'SULLIVAN, C., HOWLETT, S., MCDONNELL, R., MORVAN, Y., AND O'CONOR, K. 2004. Perceptually adaptive graphics. In *Eurographics State of the Art Reports*.

PAINTER, J., AND SLOAN, K. 1989. Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89*, ACM Press, 281–288.

PRIKRYL, J., AND PURGATHOFER, W. 1999. Overview of perceptually-driven radiosity methods. Tech. Rep. TR-186-2-99-26, Vienna, Austria.

RAMASUBRAMANIAN, M., PATTANAIK, S., AND GREENBERG, D. 1999. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 73–82.

SHIRLEY, P. 1990. A ray tracing method for illumination calculation in diffuse-specular scenes. In *Proceedings of Graphics Interface '90*, Canadian Information Processing Society, Toronto, Ontario, 205–12.

SILLION, F., AND PUECH, C. 1989. A general two-pass method integrating specular and diffuse reflection. In *SIGGRAPH '89*, ACM Press, 335–344.

SLUSALLEK, P., STAMMINGER, M., HEIDRICH, W., POPP, J.-C., AND SEIDEL, H.-P. 1998. Composite lighting simulations with lighting network. *IEEE Computer Graphics and Applications 18*, 2, 21–31.

STOKES, W. A., FERWERDA, J. A., WALTER, B., AND GREENBERG, D. P. 2004. Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. *ACM Trans. Graph. 23*, 3, 742–749.

SUNDSTEDT, V., CHALMERS, A., CATER, K., AND DEBATTISTA, K. 2004. Top-down visual attention for efficient rendering of task related scenes. In *Vision, Modeling and Visualization*.

SUNDSTEDT, V., DEBATTISTA, K., LONGHURST, P., CHALMERS, A., AND TROSCIANKO, T. 2005. Visual attention for efficient high-fidelity graphics. In *Spring Conference on Computer Graphics (SCCG 2005)*.

WALLACE, J. R., COHEN, M. F., AND GREENBERG, D. P. 1987. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *SIGGRAPH '87*, ACM Press, 311–320.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88*, ACM Press, 85–92.

WARD, G. J. 1994. The radiance lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, 459–472.

WHITTED, T. 1980. An improved illumination model for shaded display. *Commun. ACM 23*, 6, 343–349.

YEE, H., PATTANAIK, S., AND GREENBERG, D. 2001. Spatiotemporal sensitivity and Visual Attention for efficient rendering of dynamic Environments. In *ACM Transactions on Computer Graphics*, vol. 20, 39–65.